Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Previously Canceled)

2. (Previously Presented): A front end compiler system for generating code to be used by an execution environment, said front end system comprising:

a metadata module that compiles information to produce metadata information;

a code module that compiles information to produces executable instructions; and

wherein the front end compiler consumes information from a native source code file and metadata information produced by the front end compiler.

3. (Previously Presented): A front end compiler system for generating code to be used by an execution environment, said front end system comprising:

a metadata module that compiles information to produce metadata information;

a code module that compiles information to produces executable instructions; and

wherein the metadata information and executable instructions are the result of compiling a source file in a first language and the front end compiler consumes metadata information produced by a different front end compiler as a result of compiling a source file in a second language.

4. (Original) A system for compiling a computer program, the program written in a native source code language, the system comprising:

a plurality of front end compilers, wherein each front end compiler is associated with a native source code language, and wherein at least one front end compiler is associated with a different native source code language than one other front end compiler;

each front end compiler compiles common language information in addition to native source code; and

each front end compiler produces a common language file that can be consumed by a runtime environment that is targeted for a particular machine.

5. (Original) A system as defined in claim 4 wherein the runtime environment comprises:

a loader for loading the common language file into the runtime environment; and

a layout engine for examining the common language file and determining the layout for classes and objects used at runtime.

6. (Original) A system as defined in claim 5 wherein the runtime environment further comprises:

a stack walker that keeps track of a call stack during runtime; and

a garbage collector for managing memory allocation during runtime.

7. (Previously Presented): A computer readable medium having stored thereon a data structure comprising a common language file produced by a front end compiler that consumes a native source code file written in a native source code language, wherein the native source code language is one of a plurality of source languages, the common language file comprising:

a common language instructions section having instructions in a common language, the instructions related to the written program functions of the native source code file and consumed metadata, wherein the metadata describes written program functions of another native source code language file, the common language used to represent the written program functions is adapted to represent written program functions originally written in at least two different source code languages; and

a metadata portion that describes the common language instructions in the common language instructions section.

8. (Original) A common language file as defined in claim 7 wherein the common language is a common intermediate language.

9. (Previously Presented): A common language file as defined in claim 7 wherein the native source language is one of: a procedural language, an object oriented language, and a functional language.

10. (Previously Presented): A common language file as defined in claim 7 wherein the front end compiler is adapted to compile native source code files and a common language file.

11. (Previously Presented): A common language file as defined in claim 7 wherein the compiler is adapted to compile a native source code file, the native source code file utilizing a

common language library, and wherein the common language library may be consumed by different front end compilers associated with different native source code languages.

12. (Currently Amended): A computer readable medium having stored thereon a data structure comprising a generated common language file produced by a front end compiler system, wherein the front end compiler system is adapted to compile other common language files, said generated common language file comprising:

a metadata section; and

an executable instructions section, wherein the executable instruction section ~~is suitable for use with~~ comprises instructions representing a plurality of source languages.

13. (Original) A front end compiler as defined in claim 12 wherein the compiler also compiles procedural programming language files.

14. (Original) A front end compiler as defined in claim 12 wherein the compiler also compiles functional programming language files.

15. (Original) A front end compiler as defined in claim 12 wherein the compiler also compiles object oriented programming language files.

16. (Previously Presented): A computer readable medium having stored thereon a data structure comprising a common language library describing functions for use in other programming language files, the common language library represented in a common language, and wherein at least one of the programming language files is written in one native source code language and at least one other programming language file is written in a different programming language.

17. (Original) A method of compiling a computer program written in a native source language and having an import statement that imports a common language file, said method comprising:

parsing the computer program;

examining each statement during the parsing act and determining if the statement is an import statement related to the common language file;

if the statement relates to the common language file, reading the common language file into a symbol table;

if the statement relates to a native language symbol table entry, adding the information into the symbol table; and

if the statement relates to output generation, supplying the statement to a output generator.

18. (Original) A method as defined in claim 17 wherein the common language file may be imported by different source language files.

19. (Original) A method as defined in claim 17 wherein the common language file may be imported by a procedural source language file and an object oriented source language file.

20. (Original) A method as defined in claim 17 wherein the computer program is written in a procedural programming language and the common language file may be imported by an object oriented source language files.

21. (Original) A method as defined in claim 17 wherein the output generator produces a second common language file wherein the second common language file is different from the imported common language file.

22. (Original) A method as defined in claim 21 wherein the second common language file has a metadata section and a common language instructions section.

23. (Original) A method as defined in claim 22 wherein the act of reading the common library file into a symbol table further comprises reading the metadata into the symbol table.

24. (Original) A computer program product readable by a computer, the product encoding instructions for executing a computer process for compiling a source language file, wherein the source language file comprises an import statement relating to a common library file, said process comprising:

determining that the source language file has an import statement relating to a common library file;

reading information from the common library file into a symbol table; and

compiling the source language file using the symbol table.